UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Diseño e implementación de una aplicación gamificada para motivar el camino de vida de personas con TEA

Rodrigo Die Cabrera

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Diseño e implementación de una aplicación gamificada para motivar el camino de vida de personas con TEA

Autor: Rodrigo Die Cabrera
Tutor: Javier Gómez Escribano

junio 2021

Rodrigo Die Cabrera
Diseño e implementación de una aplicación gamificada para motivar el camino de vida de personas con TEA

Rodrigo Die Cabrera
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos

Agradecimientos

A las dos personas más importantes en mi vida, mi madre Martha Cabrera Sierra y mi hermana Cecilia Die Cabrera. Su compañía, esfuerzo y amor es lo que me ha hecho seguir adelante todos los días, y lo que me ha permitido estar hoy escribiendo estas palabras de agradecimiento. Gracias por vuestro apoyo en mis momentos más bajos, gracias por las risas del día a día, gracias por ser como sois y gracias por seguir acompañándome en las aventuras que nos esperan en el futuro. Y por supuesto, gracias a mi padre, Ricardo Die Dejean, por convertirse muchas veces en la voz de mi conciencia, fruto de muchos consejos y aprendizajes que no hubieran sido posibles sin él.

A toda mi familia, tanto la que me acompaña aquí en Madrid como la que me espera siempre con los brazos abiertos al otro lado del charco. En mención especial a mis abuelos, Martha Cecilia Sierra de Cabrera, Pauline Moran O'Callaghan, Luis Ernesto Cabrera Caicedo y José Die Goyanes, por toda la sabiduría y experiencia compartida.

A todos los profesores que he tenido en este largo camino, en particular a mi tutor Javier Gómez Escribano, por la oportunidad dada, por su ayuda y por haber estado siempre tan atento.

Y por último a mis amigos. Diego Vicente, un amigo con el que hablar. Pablo Martínez, un amigo al que escuchar. Miguel Agundo, un amigo que siempre estará.

Gracias.

Resumen

En este trabajo de fin de grado se ha desarrollado una herramienta software con el objetivo de motivar el camino de vida, las interacciones sociales y el aprendizaje visual de niños con Trastornos del Espectro Autista (TEA). La herramienta ofrece una experiencia gamificada que invita al usuario a un entorno tranquilo y divertido para facilitar el cumplimiento de los objetivos. Esta experiencia gamificada está basada en el juego tradicional de coleccionar cromos.

Dentro de la aplicación, los usuarios tendrán asignadas colecciones que deberán completar, las cuales pueden ser de cualquier tema. Se busca que estos temas sean del interés del alumno y estén relacionados con materias que se enseñan en el colegio. Para completar una colección, los usuarios pueden obtener nuevos cromos mediante un sistema de recompensas basado en completar hitos de su camino de vida. Estos hitos estarán marcados por el profesor. Por último, con el fin de promover la interacción social entre usuarios, estos pueden intercambiar cromos entre ellos.

Se ha decidido que la herramienta a implementar sea multiplataforma. Para ello, tras el estudio y análisis de las distintas opciones, se escogió Flutter, el framework de código abierto desarrollado por Google.

A lo largo del documento se va a describir el proceso que se ha seguido para desarrollar la herramienta planteada, distinguiendo entre las distintas fases que lo componen: análisis de requisitos, diseño del modelado de datos e interfaz del usuario e implementación del proyecto.

Palabras clave

Autismo, camino de vida, Flutter, multiplataforma, pictograma, aplicación gamificada

Abstract

In this final degree work, a software tool has been developed with the aim of motivating the life path, social interactions and visual learning of children with Autism Spectrum Disorders (ASD). The tool offers a gamified experience which invites the user to a calm and fun environment in order to facilitate the fulfillment of the objectives. This gamified experience is based on the traditional game of collecting stickers.

Within the application, users will be assigned collections to complete, which can be of any topic. The aim is for these topics to be of interest to the student and related to subjects taught at school. To complete a collection, users can obtain new stickers through a reward system based on completing milestones in their life path. These milestones will be marked by the teacher. Finally, in order to promote social interaction between users, users can exchange stickers with each other.

It was decided that the tool to be implemented should be cross-platform. For this purpose, after studying and analyzing the different options, Flutter, the open source framework developed by Google, was chosen.

Throughout the document the process followed to develop the proposed tool will be described, distinguishing between the different phases that compose it: requirements analysis, data modeling and user interface design, and project implementation.

Keywords

Autism, life path, Flutter, cross-platform, pictogram, gamified application

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Introducción

1.1. Motivación

La interacción social de una persona cobra un valor especial durante la infancia, ya que es imprescindible para desarrollar la identidad y personalidad de cada uno. Por norma general, las primeras interacciones sociales de los niños son con sus padres, a las que le siguen, a medida de que van creciendo, aquellas con sus amigos. Estas interacciones que se dan día a día, ya sea en el colegio o en el parque, les enseñan a cómo iniciar y mantener relaciones y a aceptar a las personas que son diferentes a ellos. En el caso de los niños con Trastornos del Espectro Autista (TEA), presentan dificultades de relación social, haciendo que actividades como jugar al escondite o intercambiar cromos supongan una mayor complicación.

Partiendo de esta idea tradicional del intercambio de cromos, se ha propuesto una herramienta digital con el propósito de promover la socialización entre ellos. Se trata de una aplicación para dispositivos móviles con la que los usuarios podrán coleccionar cromos e intercambiarlos con otros compañeros y conocidos. Además, a raíz de la etapa de confinamiento vivida, acciones como enviarle a tu mejor amigo una petición de intercambio de cromos y recibir una respuesta a cambio, puede ser una forma de mantener ese pequeño contacto personal tan importante.

Pese a que los niños con TEA presentan limitaciones en estas habilidades sociales, tienen a su vez grandes capacidades visuales y de memoria, que les permiten identificar fácilmente figuras, imágenes y dibujos. Aprovechando dichas capacidades, se busca también promover el aprendizaje visual a través de la aplicación. Esto es, buscando colecciones de cromos que tengan un fin didáctico, como pueden ser, por ejemplo, una colección de animales, de instrumentos musicales, o incluso, relacionados con la familia.

Asímismo, la herramienta busca motivar este proceso de aprendizaje tan importante de la infancia, con el que los niños irán descubriendo nuevas capacidades y competencias. A este proceso se le conoce también con el nombre de camino de vida.

1.2. Objetivos

El objetivo de este TFG es desarrollar una herramienta software que motive el camino de vida de personas con TEA mediante una experiencia gamificada que, permita socializar y aportar el aprendizaje visual de algunas materias.

Para ello, la herramienta debe ser multiplataforma, porque los distintos colegios y familias cuentan con distintos dispositivos, e interesa que el alumno, sea del colegio que sea, pueda manejarla desde casa y desde el mismo colegio. Este detalle ha determinado varios aspectos del proyecto como el análisis de los distintos entornos de desarrollo multiplataforma o el uso de nuevas tecnologías.

Es de gran importancia que la herramienta ofrezca una experiencia gamificada para que le motive al alumno a seguir usándola. Para ello, se ha elegido la metáfora de los cromos.

Dentro de la aplicación los alumnos podrán:

- Coleccionar cromos de distintos temas. De esta manera, si el tema es del interés del alumno y, además, está relacionado con una materia que se enseña, completar la colección supondrá un aprendizaje para él.
- Conseguir nuevos cromos de sus colecciones mediante un sistema de recompensas al completar hitos del camino de vida que marcará el profesor. Es decir, la aplicación debe contar con un mecanismo que permita asignar estos hitos de forma personalizada para obtener esa visión general de los logros del alumno.
- Intercambiar cromos con otros compañeros, con el objetivo de motivar la interacción social entre ellos.

1.3. Estructura

Ésta memoria se divide en los siguientes capítulos:

- **Capítulo 1. Introducción.** Motivación, y objetivos del proyecto.
- **Capítulo 2. Trabajo relacionado.** En este apartado se comenta el trabajo relacionado del proyecto. Esto incluye, los trabajos de investigación y las aplicaciones del ámbito de la aplicación a desarrollar y los distintos entornos de desarrollo multiplataforma que se consideraron.
- **Capítulo 3. Análisis y diseño.** En este apartado se explica el proceso de análisis y diseño realizado en el desarrollo de la aplicación. En este se incluyen los requisitos funcionales y no funcionales, los casos de uso y los diseños del modelo de la base de datos y de la interfaz de usuario.
- **Capítulo 4. Implementación.** En este apartado se muestran los comentarios sobre la implementación de la aplicación, entre los cuales se incluyen comentarios del entorno de programación empleado, de los componentes y pantallas que construyen la aplicación y del acceso a la base de datos.
- **Capítulo 5. Conclusiones y trabajo futuro.** En este último apartado se comentarán las conclusiones a las que se ha llegado tras finalizar el proyecto y los planes de futuro del mismo.

2

Trabajo relacionado

2.1. Estado del arte

A continuación, se muestran algunos trabajos de investigación centrados en establecer las experiencias gamificadas como forma de motivar las interacciones sociales y el aprendizaje en personas con TEA. También se incluyen ejemplos de aplicaciones similares al proyecto a implementar.

2.1.1. Trabajos de investigación

Lee Cadieux y Mickey Keenan en su trabajo presentado [1], estudiaron si las habilidades de comunicación de niños con TEA se podían generalizar al mundo real. Para ello, utilizaron uno de los juegos más populares del mundo, Minecraft, el cual es de particular interés entre los niños diagnosticados con este trastorno. Dentro del juego buscaron establecer un marco en el que las habilidades de comunicación social del mundo real pudiesen ensayarse en el juego, con el fin de facilitar su reproducción en un entorno real. A este marco se le dió el nombre de Social Craft. Si los investigadores o terapeutas vivieran dentro del juego, podrían presenciar y evaluar el comportamiento de los jugadores mientras interactúan entre ellos, de la misma manera de como lo harían en una situación clínica o educativa del mundo real. Los tipos de comunicación social que tendrían lugar en el entorno del juego son los mismos que se reforzarían en un entorno del mundo real, como saludar, comentar, tomar turnos y trabajar en equipo. De esta manera, estos comportamientos y datos sociales del mundo del juego podrían mejorarse y recopilarse de la misma forma en que se haría en aquellos del mundo real.

En conclusión, proponen que dentro del entorno Social Craft, los investigadores o terapeutas utilicen modificaciones mejoradas del entorno para ensayar y reforzar las actividades de comunicación social. Al mismo tiempo, se observarían las interacciones entre compañeros, lo que podría informar de un conjunto de protocolos, que a su vez podrían gamificarse en un sistema de monitoreo automatizado.

Azadeh Afrasiabi Navan y Ali Khaleghi con su trabajo [2], buscaron evaluar la efectividad de la gamificación para mejorar la calidad de la educación en niños con TEA. Para ello, se diseñó el juego Smile1, el cual se centraba en el aprendizaje de cuatro emociones básicas: felicidad, tristeza, ira y miedo. Dentro del juego, inicialmente se le enseña al niño las cuatro emociones en forma de historias

sociales y, posteriormente, se le pide que determine las expresiones faciales relacionadas con cada parte de la historia en forma de preguntas. Recalcan que en el diseño del juego, el punto más importante a tener en cuenta es que hay que hacer el juego específicamente adecuado para niños con autismo. Esto incluye:

- Una interfaz de usuario atractiva y simple que establezca un ambiente divertido y entretenido para ellos.
- Un entorno de juego relajado y no intrusivo, debido al déficit de atención y concentración que presentan.

En la figura 2.1 se puede observar dos capturas de pantalla de la aplicación. La figura 2.1(a) muestra la primera pantalla de la aplicación, en la cual se presentan las cuatro emociones con las que se va a trabajar. Tras seleccionar una de estas emociones se muestra la historia social correspondiente, como es el caso de la felicidad en la figura 2.1(b).

(a) Pantalla de inicio de la aplicación

(b) Historias sociales de la emoción felicidad

Figura 2.1: Capturas de pantalla del juego Smile1. La primera captura 2.1(a) representa la pantalla de inicio de la aplicación, en la que se muestran las cuatro emociones. La segunda captura 2.1(b) muestra las historias sociales tras haber seleccionado la emoción felicidad

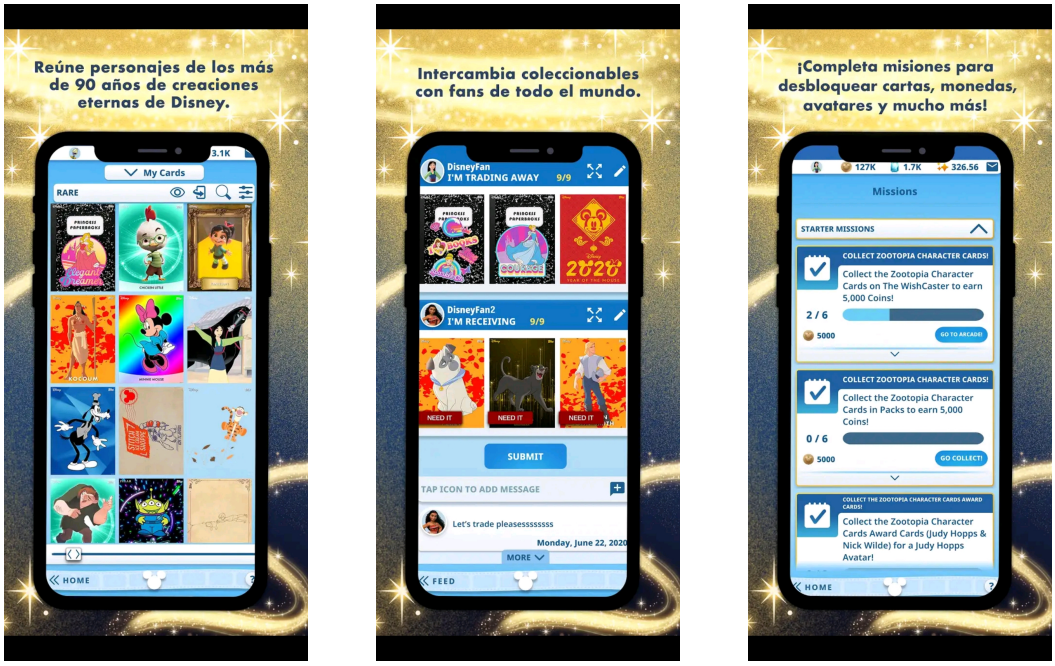
Los resultados mostraron que el uso de juegos y gamificación para motivar el aprendizaje en niños con TEA, puede tener efectos positivos en el desarrollo y promoción de sus habilidades. Tras las pruebas llevadas a cabo con el juego Smile1, los alumnos fueron capaces de aprender a detectar estas cuatro emociones.

2.1.2. Aplicaciones similares

En la práctica existen aplicaciones comerciales similares a la que se va a desarrollar. Sin embargo, no se ha encontrado ninguna que estuviese exclusivamente dirigida a personas con TEA. Las aplicaciones de colección de cromos más famosas y reconocidas son las de la compañía estadounidense Topps. [1] Como ejemplo se va a tomar la aplicación Disney Collect! por Topps. En esta aplicación los usuarios pueden coleccionar cromos de sus personajes Disney favoritos (Figura 2.2(a)). También se ofrece la posibilidad de realizar intercambios con otros usuarios (Figura 2.2(b)) y de completar misio-

[1] https://es.topps.com/

nes para desbloquear cartas y obtener puntos a cambio (Figura 2.2(b)). A diferencia de la aplicación a implementar, se nota que Disney Collect! se centra mucho más en el aspecto comercial, ya que, pese a que también cuenta con un sistema de recompensas, para hacerse con todos los cromos será necesario abrir una gran cantidad de sobres de cromos, para lo cual será necesario gastar dinero real. Otra función interesante de la aplicación, es poder convertir varios de tus cromos repetidos en uno nuevo aleatorio. El problema de esto, es que supone una alternativa al intercambio de cromos con otros usuarios, por lo que no puede ser una opción a considerar.



(a) Coleccionar cromos (b) Intercambiar cromos (c) Completar misiones

Figura 2.2: Capturas de pantalla de la aplicación Disney Collect! En cada una de las capturas se muestra una funcionalidad de la aplicación

2.1.3. Conclusiones

Como se ha podido ver en los dos trabajos de investigación comentados, los juegos y las experiencias gamificadas sirven para motivar las interacciones sociales y el aprendizaje de niños con TEA. En estos juegos, se crea un entorno relajado y divertido para ellos que les facilita el uso de las habilidades sociales propias del mundo real.

El diseño de una aplicación que va dirigida a este público tiene que estar muy cuidado. Hay varios factores que hay que tener en cuenta y que tienen que estar reflejados en la interfaz de usuario de la aplicación.

Por último, las aplicaciones basadas en coleccionar cromos más famosas están diseñadas para su

uso en un dispositivo móvil, no en un ordenador como es en el caso de los juegos empleados en los dos trabajos de investigación. Esto se debe a que el dispositivo móvil ofrece mucha más comodidad para lo que ofrece la aplicación, simulando a lo que sería el álbum de cromos tradicional.

2.2. Estado de la tecnología

En el desarrolo de aplicaciones nativas, se consigue liberar todo el potencial de los móviles y un rendimiento de primera clase, perfecto para aplicaciones que así lo necesiten. Sin embargo, implica escribir nuevo código para cada uno de los sistemas operativos con los que se quiere trabajar.
Mientras tanto, la principal ventaja en el desarrollo de aplicaciones multiplataforma es eliminar esa necesidad, manteniendo gran parte de la calidad y de las prestaciones de los lenguajes nativos. Además, de cara al programador, no es necesario conocer el lenguaje nativo de cada plataforma, lo cual supone un gran ahorro de tiempo y de energía.

A continuación, se agrupan los entornos de desarrollo multiplataforma que se tuvieron en cuenta para el desarrollo de la aplicación.

2.2.1. React Native

React Native [3] combina lo mejor del desarrollo de aplicaciones nativas con bibliotecas JavaScript, interpretando a gran velocidad el código fuente para convertirlo en los elementos nativos de cada plataforma. Es debido a esto que React Native es uno de los frameworks más utilizados a la hora de desarrollar aplicaciones multiplataforma, como es el ejemplo de Instagram, Skype o Discord.

Entre sus ventajas destacan, la función de recarga activa, muy atractiva para cualquier desarrollador ya que permite implementar cambios en el código y observar el efecto a tiempo real, y su enorme comunidad de desarrolladores en crecimiento.

Sin embargo, React Native presenta problemas de complejidad a la hora de depurar la aplicación y problemas de compatibilidad, ya que requiere el servicio de desarrolladores nativos para implementar ciertas funciones de la aplicación, como por ejemplo la cámara.

2.2.2. Ionic

Ionic Framework [4], creado en 2013, se basa en tecnologías web (HTML,CSS y JS) para desarrollar aplicaciones multiplataforma. Hasta la aparción de React Native ha sido una de las tecnologías más utilizadas para el desarrollo de este tipo de aplicaciones.

Ionic ofrece una gran lista de integraciones y *plugins*, perfectos para realizar funciones concretas.

Sin embargo, puede darse el caso en el que no se encuentre el plugin deseado y que suponga una mayor inversión de tiempo al tener que desarrollarlo uno mismo.

Como se ha comentado antes, utiliza tecnologías web (HTML, CSS y JS), lo cual implica, por una parte, que sea fácil de aprender y utilizar, ya que no es necesario aprender una nueva tecnología, y, por otra parte, que la aplicación desarrollada será más pesada, ya que habría que escribir mucho código, agregar librerías, complementos, dependencias...etc.

2.2.3. Flutter

Flutter [5] es un kit de herramientas que facilita a los desarrolladores diseñar interfaces para todo tipo de dispositivo y de tamaño de pantalla. En Flutter todo está formado por widgets. Un Widget es una forma de declarar y construir la interfaz de usuario. Flutter ofrece además un gran catálogo de widgets preconstruidos que facilitan el diseño de la aplicación, tomando prestados algunos de los conceptos del diseño web como los elementos centrados o con relleno y márgen. Esto significa que, por ejemplo, en lugar de tener que codificar un *Floating Button* como se haría en una aplicación Android y otro en una aplicación iOS, basta con utilizar el widget correspondiente, sin importar en qué plataforma se esté trabajando.

Entre las ventajas de Flutter se encuentra su rápido renderizado de vistas, el cual supera con creces a cualquiera otra solución de desarrollo móvil multiplataforma. Además, al igual que React Native, Flutter también ofrece la función de recarga activa, o *Hot Reload* [2], con la que al hacer algún cambio en el código, se podrán ver los efectos reflejados inmediatamente, sin tener que compilar la aplicación de nuevo.

El principal problema de Flutter es la necesidad de aprender una nueva tecnología, concretamente el lenguaje de programación Dart [6]. Otra desventaja es que está enfocado solo a móvil, ya que no ha salido todavía una versión enfocada para web.

2.2.4. Conclusiones

Pese a que a la hora de crear aplicaciones multiplataforma, sea React Native una de las soluciones más escogidas y más recomendadas, la posible futura implementación de una funcionalidad basada en la cámara del móvil, supuso el descarte de esta opción, ya que supondría tener que utilizar componentes y código nativos.

La idea de Ionic y sus plugins me resultó muy llamativa. Pese a esto, en mi caso personal, prefería apartarme de las tecnologías web y buscar unas nuevas, de ahí que se descartara esta opción.

En cuanto a Flutter, la necesidad de tener que aprender un nuevo lenguaje no suponía ningún pro-

[2] https://flutter.dev/docs/development/tools/hot-reload

blema, dado que lo prefería a utilizar las teconlogías web ya comentadas. Además, encontré muchas similitudes entre los plugins de Ionic y los Widgets propios de Flutter.

Debido a estas razones, se decidió que Flutter era la mejor opción para desarrollar la aplicación.

3

Análisis y diseño

En esta sección del documento se explica el proceso de análisis y diseño llevado a cabo en el desarrollo de la aplicación.

La primera parte de este proceso consiste en el estudio de los actores a los que va dirigida la aplicación. Esto se realiza con el fin de observar en que medida este público objetivo influiría en los procesos de análisis y diseño. En este caso, el público al que va dirijida la aplicación son niños con TEA. A continuación se muestra una lista de los aspectos en los que influye este público en la aplicación resultado:

- Interfaz simple y no intrusiva, que transmite un ambiento divertido.
- Color de la interfaz. Evitar algunos colores como el rojo.
- Interfaz personalizable.

Estas ideas se deben tener en cuenta a la hora de realizar el análisis y el diseño de la aplicación.

3.1. Análisis de requisitos

En esta sección se definen los requisitos del sistema que recogen la funcionalidad deseada. Esto permite mantener en un formato de listado las diferentas tareas que se llevaron a cabo, ya que toda la funcionalidad de la aplicación debe implementarse acorde a las especificaciones fijadas en esta sección.

3.1.1. Requisitos funcionales

Los requisitos funcionales describen el comportamiento del sistema cuando se cumplen determinadas condiciones, es decir, describen lo que el sistema debe hacer. Se definen mediante funciones formadas por un conjunto de entradas al sistema, un comportamiento y una salida. A continuación, se muestra la tabla de los requisitos funcionales de la aplicación, indicando para cada uno de ellos su id, su nombre, su descripción y su nivel de prioridad.

Tabla 3.1: Requisitos funcionales

ID	Nombre	Descripción	Prioridad
RF-01	Acceso estudiante	Los estudiantes acceden a la aplicación pulsando en un carta de usuario	Alta
RF-02	Ver listado de colecciones	Los estudiantes pueden ver un listado de todas las colecciones que tienen asignadas	Alta
RF-03	Ver colección	El estudiante puede ver los detalles de cada una de sus colecciones. Entre estos detalles se incluye el nombre de la colección, los cromos que la componen, el número de cromos obtenidos por el alumno y la descripción	Alta
RF-04	Ver cromo obtenido de una colección	Dada una colección asignada, el estudiante puede ver la imagen y la descripción de los cromos obtenidos	Alta
RF-05	Ver cromo no obtenido de una colección	Dada una colección asignada, el estudiante no puede ver ni la imagen ni la descripción de los cromos no obtenidos	Alta
RF-06	Obtener nuevo cromo mediante canjeo de puntos	Los estudiantes pueden obtener cromos que no poseen canjeándolos por puntos	Alta
RF-07	Ver hitos pendientes	Los estudiantes pueden ver un listado de todos sus hitos pendientes por completar. Para cada uno de estos hitos, se muestra la descripción del mismo así como la recompensa por completarlo	Alta
RF-08	Ver hitos completados	Los estudiantes pueden ver un listado de todos sus hitos completados. Para cada uno de estos hitos se muestra la descripción del mismo así como la fecha en la que se completó	Baja
RF-09	Obtener recompensa de hitos	Cuando se completa un hito, los estudiantes obtienen los puntos de recompensa correspondiente a dicho hito.	Alta
RF-10	Proponer intercambio de cromo	El estudiante puede proponer un intercambio de cromos a otro compañero	Media

Continuación en la siguiente página

Tabla 3.1 – continuación de la página anterior

ID	Nombre	Descripción	Prioridad
RF-11	Ver listado de intercambios	Los estudiantes pueden ver un listado de las peticiones de intecambio recibidas	Media
RF-12	Aceptar y completar intercambio	Si el estudiante aceptan la petición de intercambio, escogerán a continuación el cromo que desean obtener a cambio	Media
RF-13	Cambiar el color de la aplicación	Los estudiantes pueden cambiar el color primario de la aplicación desde el menú de ajustes	Baja
RF-14	Habilitar o deshabilitar los textos de la aplicación	Los estudiantes pueden habilitar o deshabilitar algunos textos de la aplicación	Baja

3.1.2. Requisitos no funcionales

Por otro lado, los requisitos no funcionales especifican como se debe comportar la aplicación, es decir, describen como funciona el sistema. Se pueden entender también como atributos de calidad del sistema. En la siguiente tabla se muestran todos los requisitos no funcionales de la aplicación.

Tabla 3.2: Requisitos no funcionales

ID	Nombre	Descripción
RNF-01	Navegabilidad	La navegación entre las distintas pantallas debe ser simple e intuitiva
RNF-02	Disponibilidad	La aplicación requiere conexión a Internet para funcionar correctamente
RNF-03	Capacidad	La ocupación de memoria debe estar optimizada. Para ello, se deberán usar imágenes de tamaño reducido
RNF-04	Compatibilidad	La aplicación estará disponible para Android y para iOS
RNF-05	Usabilidad	Los estudiantes deben de ser capaces de usar y entender toda la funcionalidad de la aplicación tras dos horas de entrenamiento

Continuación en la siguiente página

Tabla 3.2 – continuación de la página anterior

ID	Nombre	Descripción
RNF-06	Interfaz	La interfaz de la aplicación debe ser personalizable
RNF-07	Tiempo de respuesta	La carga de información no relacionada con imágenes tras un acceso a la base de datos debe tardar menos de un segundo
RNF-08	Tiempo de respuesta	La carga de imágenes tras acceder a la base de datos debe durar menos de 5 segundos
RNF-09	Diseño	El diseño de la aplicación debe recordar al de un juego educativo
RNF-10	Integridad de datos	No puede haber pérdida de información tras un error en la aplicación

3.2. Diseño de la aplicación

En la siguiente sección se explica el proceso de diseño llevado a cabo en el desarrollo de la aplicación. Primero se comentarán los casos de uso de la aplicación acompañados del diagrama correspondiente, seguido de la explicación del modelado de datos y terminando con un apartado sobre la interfaz de usuario, en el que se incluyen imágenes de los bocetos realizados.

3.2.1. Casos de uso

El objetivo principal de los casos de uso es capturar los requisitos funcionales del sistema, expresándolos desde el punto de vista del usuario. De esta manera, el cliente puede saber cómo es el sistema desde el exterior sin necesidad de entrar en los detalles de implementación. Por parte del desarrollador, supone el punto de partida del proceso de desarrollo. En la figura 3.1 se muestra el diagrama de casos de uso de la aplicación. Como se puede ver, esta cuenta con un único actor, el estudiante, del que parten todos los casos de uso.

A continuación, se describen brevemente cada uno de ellos:

- **Caso de uso 01 - Acceso estudiante**. Los estudiantes pueden acceder a la aplicación pulsando sobre su imagen.
- **Caso de uso 02 - Ver colecciones**. Los estudiantes pueden ver un listado de sus colecciones asignadas.
- **Caso de uso 03 - Ver colección**. Los estudiantes pueden ver los detalles de una colección asignada.

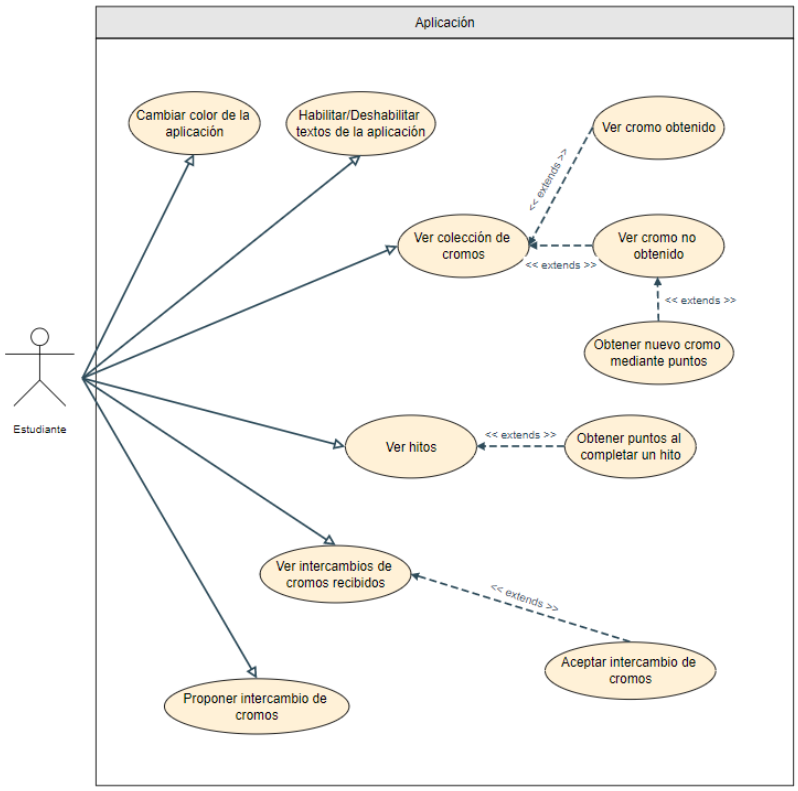


Figura 3.1: Diagrama de casos de uso

- **Caso de uso 04 - Ver cromo obtenido**. Los estudiantes pueden ver los detalles de un cromo obtenido.
- **Caso de uso 05 - Ver cromo no obtenido**. Los estudiantes no pueden ver los detalles de un cromo obtenido.
- **Caso de uso 06 - Obtener nuevo cromo mediante intercambio por puntos**. Los estudiantes pueden usar puntos para obtener un nuevo cromo.
- **Caso de uso 07 - Ver hitos**. Los estudiantes pueden ver un listado de sus hitos asignados.
- **Caso de uso 08 - Obtener puntos al completar un hito**. Los estudiantes pueden obtener puntos al completar un hito.
- **Caso de uso 09 - Proponer intercambio de cromos**. Los estudiantes pueden proponer un intercambio de cromos a otros compañeros.
- **Caso de uso 10 - Ver peticiones de intercambio recibidas**. Los estudiantes pueden ver un listado de las peticiones de intercambio que han recibido.
- **Caso de uso 11 - Aceptar intercambio**. Los estudiantes pueden aceptar una propuesta de intercambio para obtener el cromo solicitado.
- **Caso de uso 12 - Cambiar ajustes**. Los estudiantes pueden cambiar los ajustes de usuario. Entre los ajustes de usuario se incluye el color de la aplicación y habilitar o deshabilitar algunos textos de la aplicación.

La ficha de cada uno de los casos de uso se encuentra en el Anexo A de forma completa.

3.2.2. Modelado de datos

Una vez descritos todos los requisitos y casos de uso de la aplicación, es necesario estudiar que datos va utilizar la aplicación y de que forma se van a modelar. El punto de partida son los estudiantes, de los cuáles se tendrá que tener cierta información personal como puede ser su nombre, apodo e imagen. Los estudiantes pueden tener asignadas una o varias colecciones, las cuales a su vez estarán asignadas a uno o varios estudiantes. De las colecciones es necesario conocer datos como el nombre, la descripción, la imagen y, sobre todo, los cromos que la componen. De la misma manera, los cromos estarán asignados a la colección a la que pertenezcan.
Además de las colecciones, los estudiantes también tendrán asignados hitos y viceversa, con el fin de completarlos y recibir la recompensa correspondiente. Además del valor de esta recompensa, otros datos a conocer de cada hito sería su nombre y su imagen.
En cuanto a un intercambio, se verán en él siempre involucrados dos estudiantes con los dos cromos que han seleccionado para intercambiar.
Y, por último, dado que los usuarios pueden tener preferencias de configuración, estas deberán ir ligados a cada uno de ellos.

Estas relaciones entre datos quedan indicadas de manera más visual en el esquema de la figura 3.2. Como se puede observar, el resultado es un esquema simple, ya que el número de relaciones no es muy eleveado, dichas relaciones no son excesivamente complejas y la mayoría de ellas parten de un mismo punto, el estudiante.

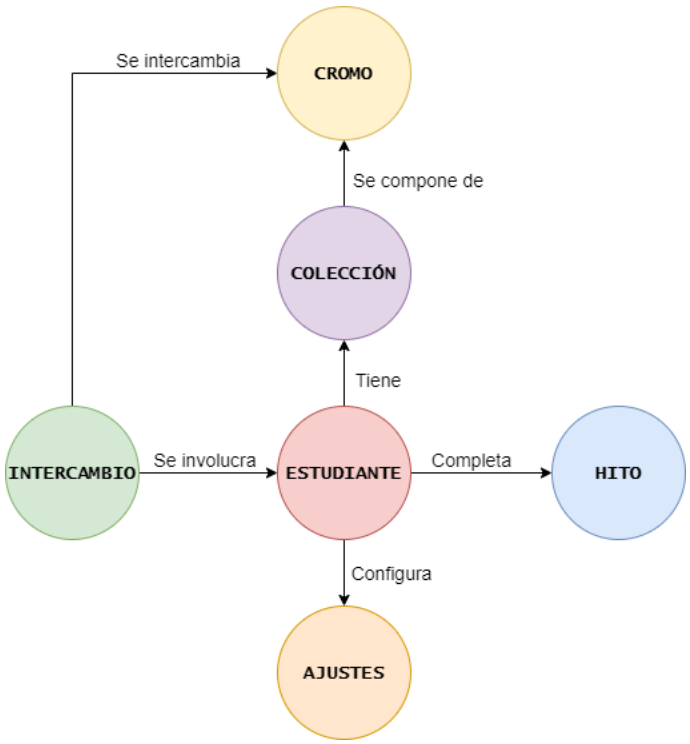


Figura 3.2: Modelado de datos

3.2.3. Diseño de la interfaz del usuario

El diseño de la interfaz de usuario va ligado al público objetivo de la aplicación. Es por ello, que se ha intentado buscar un estilo sencillo, plano, amigable y, sobre todo, intuitivo, ya que la máxima prioridad es que el estudiante pueda entender la funcionalidad de cada elemento que se muestra por pantalla. Se han realizado bocetos de cada una de las pantallas de la aplicación a través de la herramienta Moqups [1], cuya función de añadir comentarios ha sido de gran ayuda para discutir opiniones con el profesor así como para mantener constancia de los cambios que se fueron realizando en el tiempo. A continuación, se explicarán la última versión de estos bocetos, tras la cual se acordó el diseño general de la aplicación que se llevaría luego a implementación.

3.2.4. Acceso de usuario

Tras ingresar en la aplicación, los usuarios accederán a la pantalla Acceso de usuario, mostrada en la figura 3.3. En esta pantalla se muestra para cada uno de los alumnos, una pequeña sección circular con su apodo y con su imagen. De esta forma, el alumno podrá acceder a la aplicación simplemente pulsando en su imagen. Por pantalla se mostraran hasta 6 alumnos, con scroll horizontal por si el número fuera mayor.

Figura 3.3: Boceto de la pantalla Acceso de usuario

3.2.5. Menú principal

El menú principal cuenta con tres botones personalizados para acceder a la funcionalidad de la aplicación:

- Acceder a las colecciones del alumno.
- Acceder a los hitos del alumno.

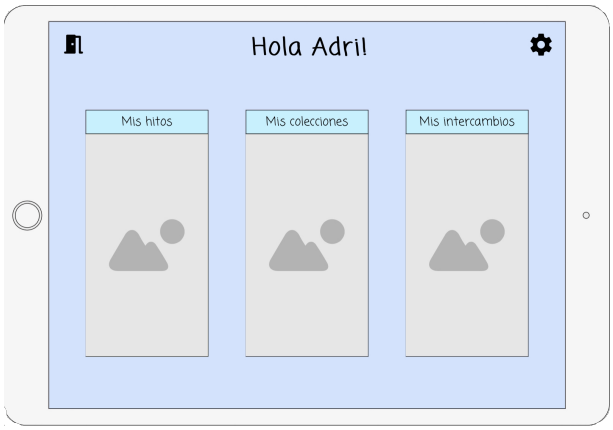
[1] https://moqups.com/

- Acceder a los intercambios del alumno.

La disposición de estos botones es siempre la misma, no cambia nunca. Además, en la esquina superior derecha de esta pantalla, hay un botón para acceder a los ajustes de usuario. El boceto de esta pantalla queda indicado en la figura 3.4(a).

3.2.6. Ajustes de usuario

Los ajustes de usuario se muestran en dos columnas como se puede observar en la figura 3.4(b). En la primera de ellas, el usuario puede seleccionar uno de los colores disponibles para cambiar el color de la aplicación, y, en la segunda, puede marcar o desmarcar la opción de mostrar descripciones de colecciones y cromos.



(a) Boceto de la pantalla Menú principal

(b) Boceto de la pantalla Menú de ajustes

Figura 3.4: Boceto del menú principal (figura 3.4(a)) y del menú de ajustes (figura 3.4(b))

3.2.7. Mis colecciones

En la figura 3.5(a) se muestra la pantalla a la que accede el usuario tras seleccionar la opción Mis colecciones desde el menú principal. Para cada colección del alumno, se mostrará la tarjeta de colección correspondiente, la cual muestra la siguiente información:

- Nombre de la colección.
- Número de la colección.
- Imagen de la colección.
- Número de cromos obtenidos por el alumno/Número total de cromos de la colección.

Cuando el alumno pulse sobre una de estas tarjetas, navegará a la pantalla de la colección correspondiente.

3.2.8. Mi colección

En la pantalla correspondiente a la colección seleccionada por el alumno, la información que se muestra es la siguiente:

- Cromos que componen la colección, a través de un scroll horizontal de tarjetas de cromo.
- Descripción de la colección.

Para cada cromo de la colección, se muestra una tarjeta de cromo compuesta de su imagen, su nombre y un color que representa su rareza. En función de si el alumno posee o no dicho cromo, la imagen será visible o no. Si el alumno ha desactivado la opción de mostrar descripciones desde la pantalla de ajustes, el propio listado de los cromos de la colección ocuparan toda la pantalla. Cuando el alumno pulse sobre una de estas tarjetas, navegará a la pantalla del cromo correspondiente. El boceto de esta pantalla queda indicado en la figura 3.5(b).

3.2.9. Cromo obtenido

Si el alumno pulsa sobre la tarjeta de un cromo que ya posee, se mostrará por pantalla dos columnas con la siguiente información:

- La imagen del cromo (de mayor tamaño que en la pantalla de colección).
- Descripción del cromo.

Si el alumno ha desactivado la opción de mostrar descripciones desde la pantalla de ajustes, la imagen del cromo pasará a ocupar el centro de la pantalla. Véase la figura 3.6(a).

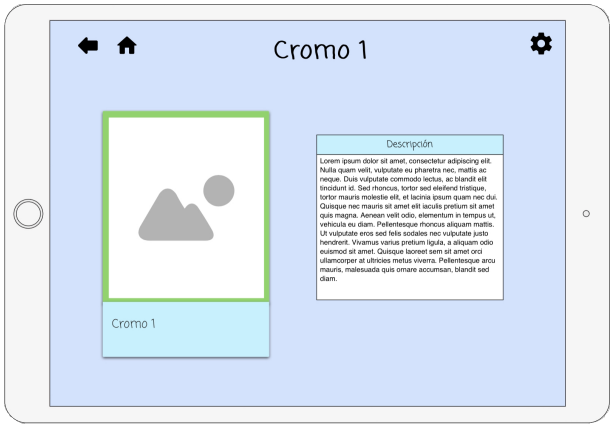
(a) Boceto de la pantalla Mis colecciones

(b) Boceto de la pantalla Mi colección

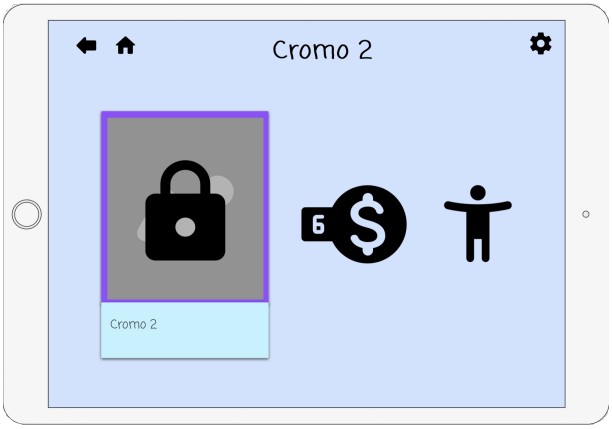
Figura 3.5: Bocetos de las pantallas relacionadas con la gestión de colecciones

3.2.10. Cromo no obtenido

Si el alumno pulsa sobre la tarjeta de un cromo que no posee, se mostrará por pantalla las dos opciones que tiene el alumno para obtenerlo, tal y como se presenta en la figura 3.6(b). La primera de ellas, intercambio por puntos, está representada por un símbolo que transmita al alumno que va a adquirir un cromo a cambio de algo. Algunos ejemplos pueden ser el símbolo del dólar o del euro o el símbolo del carrito de la compra. Tras seleccionar esta opción, se muestra por pantalla un mensaje de confirmación que indica además la cantidad de puntos necesarios para realizar el intercambio. La segunda opción, intercambio con otro alumno, está representada también por el pictograma correspondiente, el cual transmita al alumno que va a hacer un intercambio con otra persona, es decir, que indique al alumno que va a haber otra persona involucrada. Tras seleccionar esta opción, el alumno deberá seleccionar el compañero con el que quiere realizar el intercambio. Para ello, se puede emplear la misma pantalla de Acceso de usuario. Tras seleccionar el alumno, este podrá ver la petición de intercambio recibida en la pantalla correspondiente.



(a) Boceto de la pantalla Cromo obtenido



(b) Boceto de la pantalla Cromo no obtenido

Figura 3.6: Bocetos de las pantallas relacionadas con la gestión de cromos

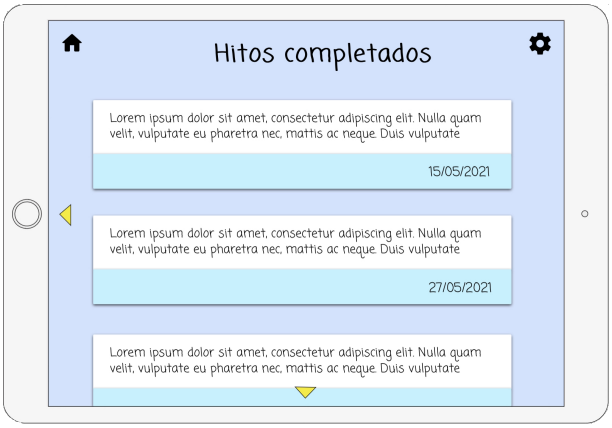
3.2.11. Mis hitos pendientes

En la imagen 3.7(a) se muestra la pantalla a la que accede el usuario tras seleccionar la opción Ver mis hitos desde el menú principal. Esta se compone del listado de hitos pendientes por completar del alumno. Para cada uno de ellos, se indicará la descripción del mismo y la recompensa en puntos que obtendrá el alumno al completarlo. Tras completar un nuevo hito, se mostrará por pantalla un aviso indicando al alumno la recompensa obtenida.

3.2.12. Mis hitos completados

En la misma pantalla de Mis hitos pendientes, el usuario puede hacer scroll horizontal para acceder a la página de los hitos ya completados, tal y como se indica en la imagen 3.7(b). En este caso, en lugar de la recompensa se muestra la fecha en la que el alumno completó el hito. Ambos listados son navegables a través de un scroll vertical.

(a) Boceto de la pantalla Hitos pendientes



(b) Boceto de la pantalla Hitos completados

Figura 3.7: Bocetos de las pantallas relacionadas con la gestión de hitos

3.2.13. Mis intercambios

En la figura 3.8 se muestra la pantalla a la que accede el usuario tras seleccionar la opción de Ver mis intercambios desde el menú principal. Esta se compone del listado de peticiones de intercambio que ha recibido el usuario. Cada petición está representada por la siguiente información:

- La imagen del cromo que quiere obtener el compañero que propuso el intercambio.
- La imagen y nombre del compañero que propuso el intercambio.

Cada petición es accesible a través de un scroll horizontal y puede ser aceptada o rechazada por el alumno. En el caso de ser aceptada, el alumno debe elegir el cromo que desea obtener con el intercambio. Para ello, se muestra un listado de los cromos que no posee de la misma forma que en la pantalla de colección, sin incluir el apartado de descripción.

Figura 3.8: Boceto de la pantalla Mis intercambios

4

Implementación

Una vez finalizada la fase de análisis y diseño de la aplicación, se da comienzo a la fase de desarrollo, en la que se trasladan a código todas las ideas obtenidas anteriormente. En esta sección se comentan todos los pasos seguidos en esta fase.

El primero de ellos fue el estudio de la anotomía de un proyecto Flutter, a raíz del cual se empezarían a escribir las primeras líneas de código. Como ya se comentó anteriormente, todo dentro de una aplicación Flutter es un widget, cada uno con su funcionalidad propia.

A cada uno de estos widgets se les puede ver como piezas de Lego, pudiéndolas conectar y personalizar de mil formas distintas obteniendo así siempre un resultado nuevo. Para que se entienda mejor esta metáfora, se incluye en la figura 4.1 el ejemplo del widget Container. Como se puede observar, este cuenta con un gran número de atributos, los cuales se pueden ir cambiando para obtener el contenedor deseado, es decir, nuestro contenedor personalizado. De entre todos ellos destaca *child*, ya que es en este atributo donde se va a indicar cuál es el widget hijo de nuestro contenedor. De esta forma, se están conectando dos widgets, el contenedor padre y su hijo, que, a su vez, podrá tener uno, más de uno o ningún hijo.

```
package:flutter/src/widgets/container.dart
(new) Container Container({Key key,
                           AlignmentGeometry alignment,
                           EdgeInsetsGeometry padding,
                           Color color,
                           Decoration decoration,
                           Decoration foregroundDecoration,
                           double width,
                           double height,
                           BoxConstraints constraints,
                           EdgeInsetsGeometry margin,
                           Matrix4 transform,
                           AlignmentGeometry transformAlignment,
                           Widget child,
                           Clip clipBehavior = Clip.none})

Containing class: Container
```

Figura 4.1: Atributos del widget Container

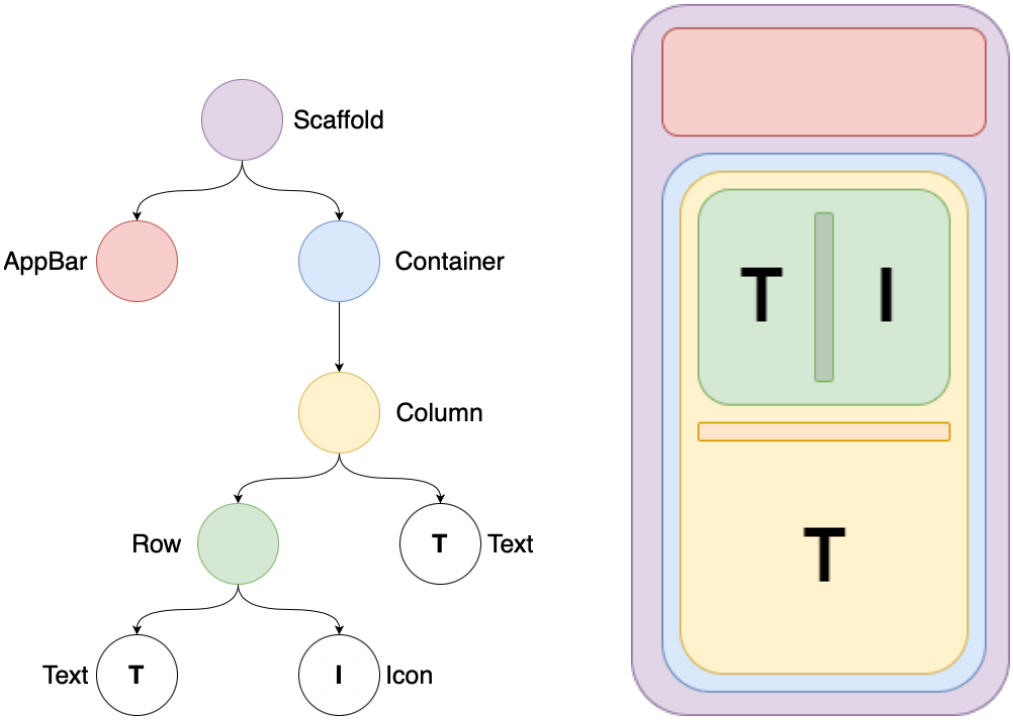
Sin embargo, este ejemplo se queda todavía corto para entender la anatomía de un proyecto Flutter. Para ello, se muestra a continuación otro ejemplo algo más complejo. Lo primero que se puede hacer es crear un Scaffold, o andamio en español. Este widget representa una pantalla en blanco de nuestra aplicación, dentro de la cual se van a añadir otros elementos que puedan interesar, como, por ejemplo, una barra de aplicaciones en la parte superior de la pantalla, empleando el widget AppBar y un contenedor para almacenar dentro de él el contenido de la aplicación, empleando el ya mencionado widget Container. Este contenedor tendrá como hijo el widget Column que, a su vez, albergará otros widgets hijo en su interior, con la particularidad de que estos estarán alineados en vertical, en forma de columna. Por ejemplo, estos pueden ser el widget Row, cuyos widgets hijo se alinearán en forma de fila y el widget Text, que representa un texto. Por último, dentro de Row se incluye otro widget Text y el widget Icon, que representa un icono.

Tras finalizar la conexión entre todos estos widgets, se obtiene el árbol de widgets resultado, como el que se muestra en la figura 4.2(a). En él se puede observar que cada uno de los widgets representa un nodo del árbol, siendo el Scaffold el nodo raíz del que parten el resto de nodos. Salvo alguno de ellos, como Text y Icon en nuestro ejemplo, los widgets pueden tener uno o incluso varios widgets hijo, lo que resulta en un aumento de la profundidad del árbol. Sin embargo, que un widget pueda tener hijos no significa que deba tenerlos por obligación, como es el caso del nodo rojo AppBar, el cual acompañado de los widgets Text y Icon representan los nodos hoja del árbol. Por otra parte, la figura 4.2(b) muestra la que sería la distribución de estos widgets en la pantalla de un dispositivo móvil cualquiera. Como se puede observar, todos los widgets viven dentro unos de otros como si fueran piezas que vamos colocando unas encima de otras.

Una vez entendida la anatomía de un proyecto Flutter, se comenzó con la implementación de la aplicación. La idea fue implementar todas las pantallas con datos hardcodeados para, a continuación, implementar el acceso a la base de datos en la aplicación y pintar las pantallas con datos reales. Sin embargo, tras entender como funciona Flutter, se decidió que el proceso de implementación de las pantallas no iba a consistir en ir implementando una en una. En cambio, una mejor idea era implementar primero componentes independientes que luego se reutilizarían en distintas pantallas. De esta manera, se ahorraría así tiempo y trabajo. Y, por último, una vez pintadas las pantallas con datos reales, se terminaría de implementar toda la lógica necesaria.

4.1. Componentes

Cada componente representa entonces, un conjunto de widgets anidados entre sí que se reutilizan en distintas pantallas. Por ejemplo, volviendo al ejemplo de la figura 4.2, imaginemos un componente al que llamaremos Etiqueta, el cual estará formado por el nodo verde y sus dos hijos, es decir, por el Widget Row y sus hijos Text y Icon. Ahora, como si de una clase se tratara, el componente Etiqueta tendrá su constructor propio, al que se le pasarán todos los elementos que requiera. Una posibilidad



(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.2: Ejemplo ilustrativo de la anotomía de un proyecto Flutter sencillo

sería pasarle la cadena que mostrará su hijo Text y el icono que mostrará el hijo Icon, como se puede observar la figura 4.3.

```
class Etiqueta extends StatelessWidget {
  Etiqueta({@required this.myText, @required this.myIcon});

  final String myText;
  final IconData myIcon;
```

Figura 4.3: Código - Constructor del componente etiqueta

De esta forma, cada vez que se quiera crear el componente Etiqueta, bastaría con indicar estos dos elementos sin necesidad de repetir código. En este ejemplo concreto no sería tanto el ahorro de tiempo y código, pero cuando se tienen varios componentes y de mayor tamaño, como es el caso, la diferencia es muy notable.

Se van a diferenciar entre dos tipos de componentes: básicos y complejos. Los componentes básicos no tienen mucho sentido por si solos, pero son imprescindibles en la construcción de los componentes más complejos. Dentro de los componentes básicos se encuentran: Reusable Container, Reusable Card, Column Card y Student Avatar.

A continuación, se indican la función y el árbol de widgets de cada uno de los componentes imple-

mentados. Además, para los componentes complejos, el árbol de widgets vendrá acompañado de una imagen con la disposición de todos los widgets en el dispositivo.

4.1.1. Reusable Container

Reusable Container está formado únicamente por el widget Container, con la diferencia de que en este caso el estilo de sus bordes será circular por defecto. De esta manera, se mantiene el mismo diseño de bordes en todos los componentes que utilizan Reusable Container. En el constructor se incluyen además los valores de los márgenes, el color del contenedor y, si fuese el caso, el widget hijo que alberga.

Como se trata de un único widget no se ha incluido árbol de widgets en este caso.

4.1.2. Reusable Card

Este componente es de los más utilizados en toda la aplicación, ya que es la base de un conjunto de componentes que adoptarán un diseño parecido al de un cromo. Como se muestra en la figura 4.4, tiene como hijo el widget GestureDetector, el cual, a través de su atributo *onTap*, indica a su hijo, Reusable Container en este caso, la función que debe hacer después de que el usuario pulse sobre él. A su constructor se le debe indicar un color y, si se diera el caso, el que será el hijo de Reusable Container además de la función que realizaría tras ser pulsado por el usuario.

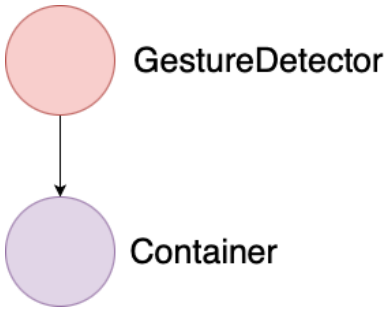


Figura 4.4: Árbol de widgets del componente Reusable Card

4.1.3. Column Card

La idea de este componente, cuyo árbol de widgets se puede observar en la figura 4.5, es listar a sus hijos verticalmente haciendo uso de Column. Estos hijos son siempre widgets Expanded, los cuales tienen la peculiaridad de tender a expandirse hasta ocupar todo el espacio que les sea posible. Si se quisiera usar varios más de un widget Expanded, como es el caso, el espacio disponible se divide entre ellos de acuerdo con su factor de flexibilidad, el cual queda indicado a través de su atributo *flex*.

Al constructor se le pasará entonces, para cada uno de los hijos Expanded, su factor de flexibilidad y el que será su widget hijo. Es por eso que en la imagen los hijos de ambos Expanded aparecen con línea discontinua, ya que pueden ser cualquier cosa pero a su vez, es obligatorio que existan.

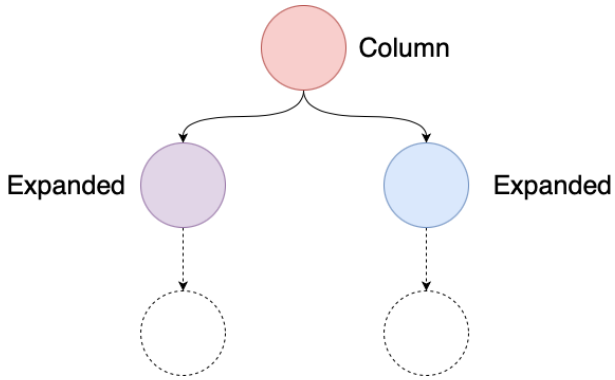


Figura 4.5: Árbol de widgets del componente Column Card

4.1.4. Student Avatar

Como se indica en el árbol de widgets de la figura X, este componente cuenta con el widget CircleAvatar, el cual consiste en un circulo que representa a un usuario. Este se utiliza dos veces ya que el primero sirve como marco del segundo y este segundo contiene la imagen del estudiante. Para crear este componente será necesario indicar su color, el radio del segundo CircleAvatar y la imagen del estudiante.

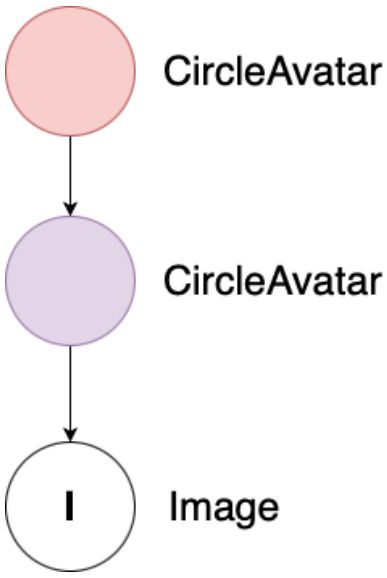
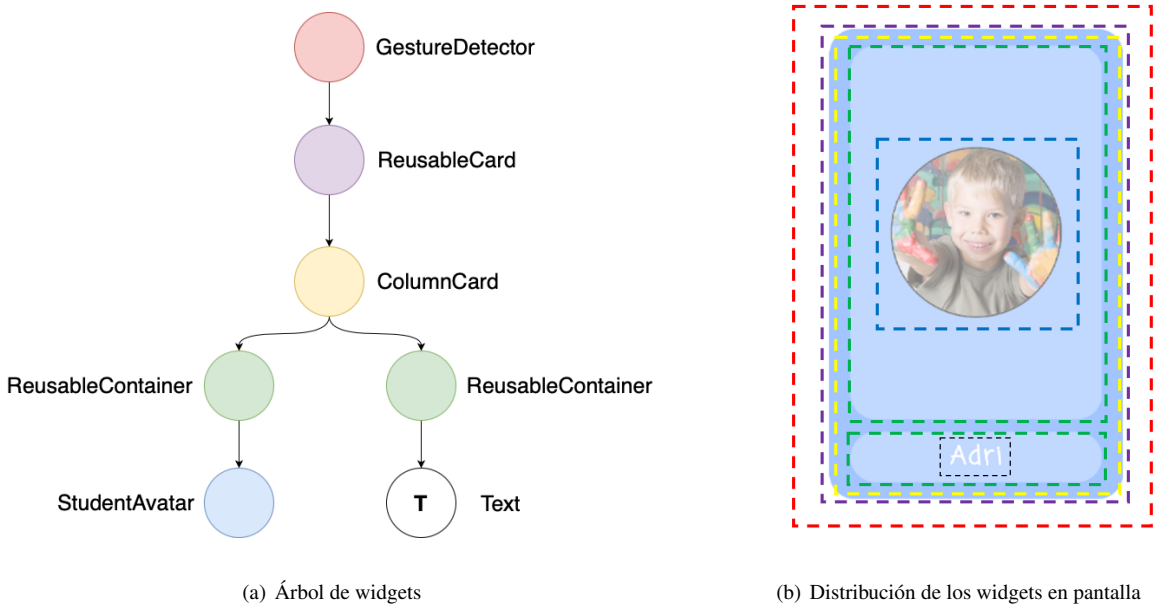


Figura 4.6: Árbol de widgets del componente Student Avatar

4.1.5. Avatar Card

En esta carta se muestra la siguiente información relativa a los estudiantes: foto de perfil y apodo. De esta manera, será de gran ayuda en aquellas pantallas en las que sea necesario elegir un usuario en concreto, como es, por ejemplo, la de acceso de usuario. En la imagen 4.7 se muestra el árbol de widgets de Avatar Card, en el que se observa que cuenta con los cuatro componentes básicos ya comentados. Para crear un Avatar Card será necesario indicar en el constructor los colores de los dos Reusable Container y la información relativa al usuario.



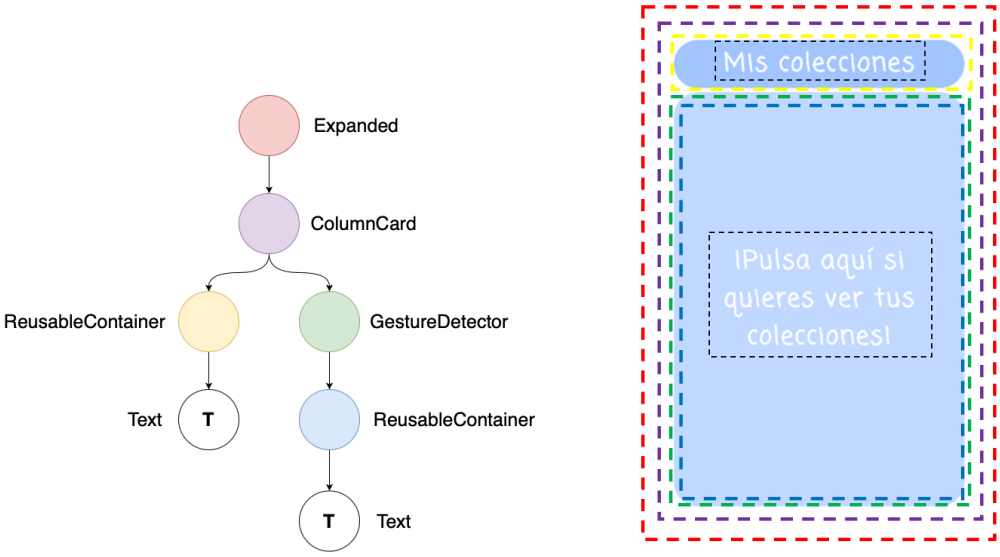
(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.7: Árbol de widgets (Figura 4.7(a)) y su disposición en pantalla (Figura 4.7(b)) del componente Avatar Card

4.1.6. Main Menu Button

Los tres botones personalizados del menú principal también hacen uso de alguno de los componentes ya comentados, tal y como se puede observar en la figura 4.8. Cada botón comienza con un pequeño contenedor en el que se indica su título, seguido de otro en el que se indica para que sirve. Al constructor se le pasa entonces, el color del botón, y los textos de ambos contenedores. Además, como se trata de un botón, se le pasará también la función *onTap* a realizar después de que el usuario pulse sobre él. En este componente se vuelve a emplear el widget ColumnCard, el cual, en este caso, tiene como nietos los dos contenedores que se acaban de comentar con factores de flexibilidad 1 y 8 correspondientemente. De esta manera, el espacio disponible del botón se divide con esta relación para los dos componentes.



(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

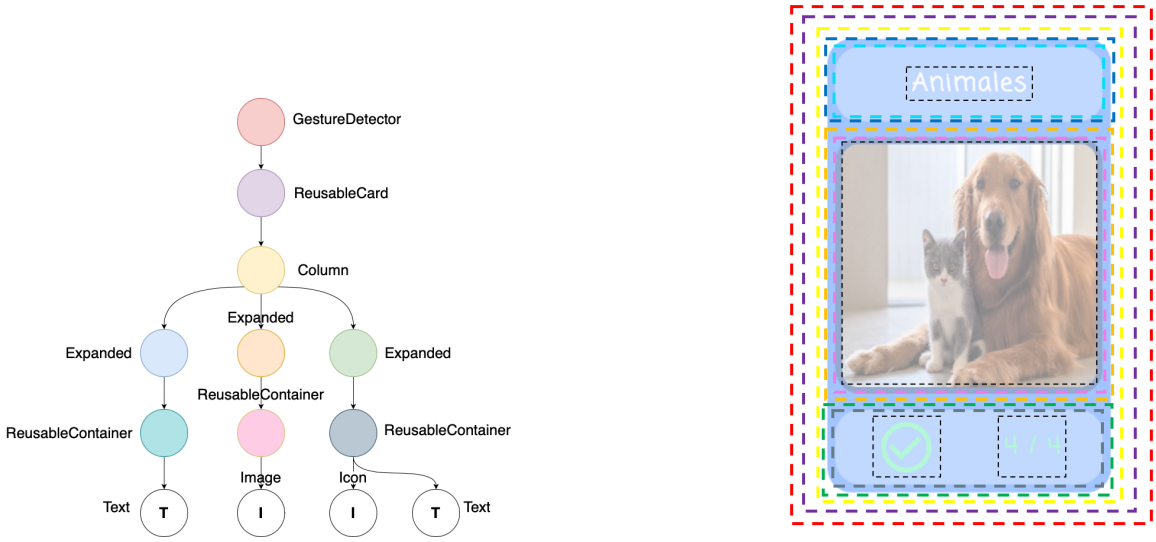
Figura 4.8: Árbol de widgets (Figura 4.8(a)) y su disposición en pantalla (Figura 4.8(b)) del componente Main Menu Button

4.1.7. Collection Card

Este componente se utiliza en la pantalla de colecciones del usuario para mostrar la información de cada una. El árbol de Widgets de este componente se muestra en la imagen 4.9(b). De este árbol es importante recalcar que su nodo raíz es un GestureDetector, ya que el usuario pulsará en una de estas cartas para navegar a la pantalla de dicha colección, y la existencia del componente Collection Progress, utilizado simplemente para mostrar en una fila el número de cromos que posee el alumno frente al número de cromos total de la colección. Para crear cada uno de estos componentes será necesario indicar los tres tonos de color que va a utilizar y la información relativa a la colección, entre la que se incluye su id, su nombre, su imagen, el número total de cromos que la componen y el número de cromos que ya posee el usuario.

4.1.8. Sticker Preview Card

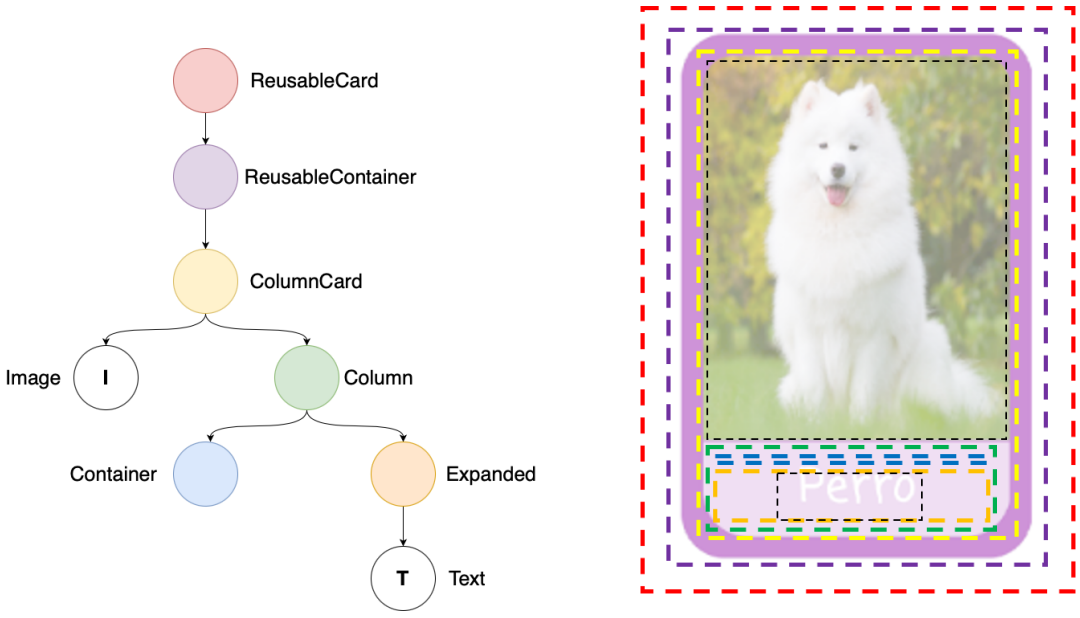
Tras elegir una colección del listado, se muestran por pantalla una vista anticipada de los cromos a través del componente Sticker Preview Card, el cual hace uso de nuevo del componente Column Card, tal y como queda indicado en la figura 4.10. En el constructor encontramos información de la colección como su nombre y descripción y la información de cada uno de los cromos que la componen, entre la que destaca su grado de rareza, y si el usuario actual la posee o no, ya que, como se ha indicado anteriormente, la imagen mostrada en esta vista cambia en función de si el usuario tiene el cromo o no.



(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.9: Árbol de widgets (Figura 4.9(b)) y su disposición en pantalla (Figura 4.9(b)) del componente Collection Card



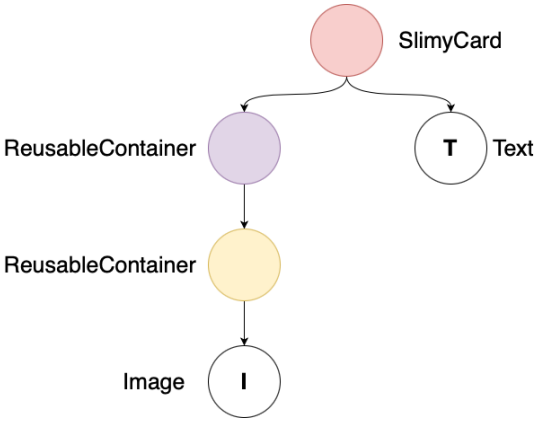
(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.10: Árbol de widgets (Figura 4.10(a)) y su disposición en pantalla (Figura 4.10(b)) del componente Sticker Preview Card

4.1.9. Owned Sticker

Este componente es un caso especial, ya que se utiliza el paquete Slimy Card [1], aportado por otro desarrollador. Además de este nuevo widget, en la imagen 4.11 se muestran el resto de Widgets que forman este componente. Por último, este componente requiere que se le indique el color a utilizar así como la información del cromo seleccionado por el usuario.



(a) Árbol de widgets

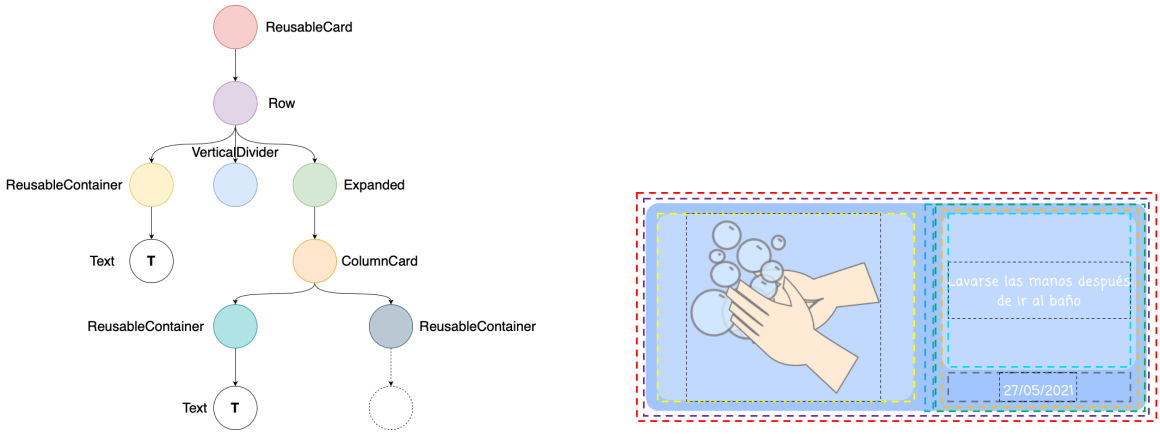
(b) Distribución de los widgets en pantalla

Figura 4.11: Árbol de widgets (Figura 4.11(a)) y su disposición en pantalla (Figura 4.11(b)) del componente Owned Sticker Card

4.1.10. Milestone Card

Entre un hito pendiente por completar y uno completado, la información que se muestra por pantalla es la misma salvo por un elemento. Es por ello, que se ha implementado un único componente al cual se le indicará cuál de los dos elementos va a mostrar. El árbol de Widgets de este componente queda indicado en la figura 4.12. Observando el árbol, es al segundo hijo de Column Card a quien se va indicar cual va a ser su hijo. En el caso de ser un hito por completar, su hijo será la recompensa por completarlo y, si es un hito completado, su hijo será un texto con la fecha en la que se completó. En el constructor de este componente se indicará su color y la información del hito correspondiente, entre la que se incluye si es un hito completado o no.

[1] https://pub.dev/packages/slimy$_card$



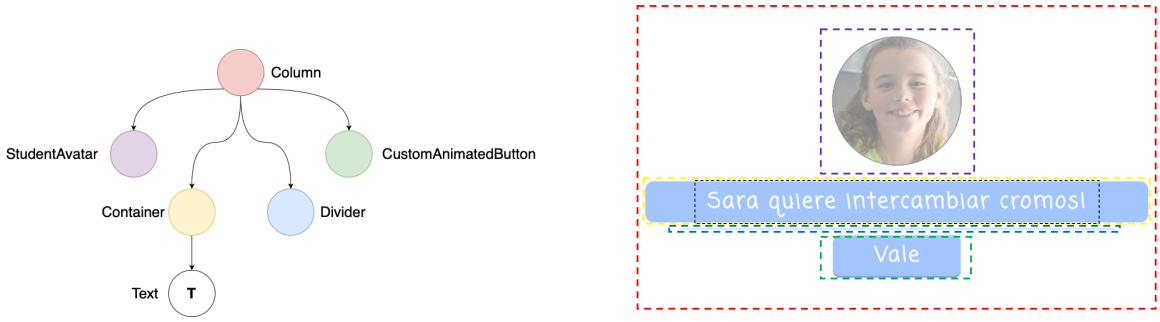
(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.12: Árbol de widgets (Figura 4.12(a)) y su disposición en pantalla (Figura 4.12(b)) del componente Milestone Card

4.1.11. Exchange Card

Este último componente se utiliza para mostrar la información de las peticiones de intercambio que ha recibido el usuario. En la figura 4.13 se observa que está compuesto de algún componente y Widget ya visto, con las novedades de los Widgets Divider, que, tal y como lo indica el nombre, se trata de una línea que separa contenidos, y CustomAnimatedButton, que se trata de un botón con animación al pulsarlo. Como en otros casos, basta con indicar el color y la información correspondiente para crear el componente. En este caso se trataría de la información de la petición de intercambio.



(a) Árbol de widgets

(b) Distribución de los widgets en pantalla

Figura 4.13: Árbol de widgets (Figura 4.13(a)) y su disposición en pantalla (Figura 4.13(b)) del componente Exchange Card

4.2. Base de datos

Se decidió que el acceso a la base de datos se haría a través de la nube, siendo necesario por lo tanto estar conectado a Internet. Esta decisión se debe a que se ha buscado evitar situaciones que pudieran confundir a los usuarios, como por ejemplo, que tras un día de juego con la aplicación en el colegio, acceden a través de otro dispositivo pero esta vez sin conexión a Internet, con la sorpresa de que los cromos que habían obtenido hace apenas horas ya no aparecen.

Para cumplir con este objetivo se decidió utilizar Cloud Firestore de Firebase [7]. Esta base de datos NoSQL mantiene los datos sincronizados mediante el uso de objetos de escucha en tiempo real. Además, ofrece la integración con otros productos Firebase, lo cual fue de gran importancia para el uso de imágenes de la aplicación.

4.2.1. Acceso a la base de datos

A continuación, se muestra la implementación del acceso a la base de datos a través de Flutter. Como ejemplo se utilizará la página Acceso de usuario. Para comenzar este proceso, es necesario primero entender la función del widget Stream. Stream proporciona una forma de recibir una secuencia de eventos, los cuales pueden ser eventos de datos o, si ha habido error, eventos de error. Tras emitir todo su evento, este notifica al oyente de que ha finalizado, el cual, tras finalizar la escucha, recibe un objeto que proporciona dichos eventos. Una vez conocida la clase Stream, es necesario introducir también el widget StreamBuilder. Este se construye a sí mismo basándose en la última *snapshot* de interacción con un Stream. En otras palabras, el widget se irá construyendo a partir del Stream indicado en su atributo *stream*. Para el caso de la pantalla tomada como ejemplo, se indicaría uno con las *snapshots* de los datos en nube de los usuarios.

Una vez indicado el Stream, se indica la estrategia de construcción en el atributo *builder*. De esta manera, se le puede indicar al widget la forma en que debe construirse dadas ciertas circunstancias, como pueden ser: si ha habido o no un error o si todavía se está estableciendo la conexión, véase la figura 4.14. En el caso de error, se muestra un mensaje de error por pantalla, en el caso de estar esperando la conexión, se muestra un indicador de progreso y en el caso de no error, se crean los objetos de usuario correspondientes.

Por último, una vez creada la lista de objetos User con todos los usuarios, se envía a la clase AvatarCardList. Esta clase cuenta con el widget ListView, el cual nos permite juntar varios widgets en una lista desplazable y organizarlos linealmente. De la misma forma que con StreamBuilder, a ListView se le puede indicar una estrategia de construcción a través de su atributo *itemBuilder*, véase la figura 4.15.

```
child: StreamBuilder<QuerySnapshot>(
  stream: _firestore.collection("users").snapshots(),
  builder:
      (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
    if (snapshot.hasError) return Center(child: Text("Error"));

    if (snapshot.connectionState == ConnectionState.waiting) {
      return Center(child: CircularProgressIndicator());
    } else {
      final users = snapshot.data.docs;
      List<User> studentsList = [];

      for (var user in users) {
        final student = User(
          id: user.id.toString(),
          name: user.data()["name"],
          profilePic: user.data()["image"],
        );  // User

        studentsList.add(student);
      }

      return AvatarCardList(users: studentsList);
    }
  },
),  // StreamBuilder
```

Figura 4.14: Código - Acceso a la base de datos desde la pantalla Acceso de usuario

```
return ListView.builder(
  scrollDirection: Axis.horizontal,
  itemBuilder: (context, index) {
    return AvatarCard(
      userId: users[index].id,
      username: users[index].name,
      profilePic: users[index].profilePic,
      outerCardColor: Colors.blueAccent,
      insideCardColor: Colors.blueAccent[100],
    );  // AvatarCard
  },
  itemCount: users.length,
);  // ListView.builder
```

Figura 4.15: Código - Construcción del widget ListView para la lista de AvatarCard

4.2.2. Gestión de imágenes en base de datos

Otro aspecto importante a la hora de implementar la base de datos es la gestión del gran número de imágenes que va a utilizar la aplicación. Como ya se ha comentado en esta sección, Cloud Firestore ofrece la integración con otros productos Firebase. Uno de ellos, Cloud Storage [8], serviría para almacenar todas estas imágenes. Se debía buscar ahora una forma de acceder a las imágenes almacenadas en Cloud Storage desde Cloud Firestore. La solución encontrada consiste en guardar en la base de datos de Cloud Firestore una cadena con la ubicación de almacenamiento de la imagen en Cloud Storage.

Continuando con el ejemplo de la pantalla Acceso de usuario del apartado anterior, como se necesita acceder a la imagen de perfil del alumno, se guardaría en el campo imagen del alumno la cadena de la ubicación de almacenamiento de dicha imagen. Para ello, se ha creado la clase FireStorageService con la función *loadImage*, la cual se encarga de descargar una imagen dada su ubicación, véase la figura 4.16.

```
class FireStorageService extends ChangeNotifier {
  FireStorageService();
  static Future<dynamic> loadImage(BuildContext context, String url) async {
    return await FirebaseStorage.instance.ref().child(url).getDownloadURL();
  }
}
```

Figura 4.16: Código - Clase FireStorageService

A continuación, en el método *build* del widget o componente correspondiente, Student Avatar en el ejemplo tomado, se añade la función *getImage* la cual devolverá un objeto Future, como se muestra en la figura 4.17. Esta clase representa un cálculo retrasado, siendo en este caso, la imagen que se obtendrá de la función *loadImage*.

```
Future<Widget> _getImage(BuildContext context, String imageUrl) async {
  Image image;
  await FireStorageService.loadImage(context, imageUrl).then((value) {
    image = Image.network(
      value.toString(),
      fit: BoxFit.scaleDown,
    );
  });

  return image;
}
```

Figura 4.17: Código - Método getImage

Por último, empleando el widget FutureBuilder y las funciones recién comentadas, se realiza el acceso a la base de datos con el fin de descargar la imagen deseada, véase la figura 4.18. A ello se le añade un control de errores muy parecido al ya comentado con StreamBuilder como ejemplo:

- Si la conexión se ha establecido y la imagen se ha descargado sin problemas, se muestra la propia imagen. En caso contrario se muestra una imagen por defecto almacenada en el propio proyecto.
- Si la conexión está en espera se muestra un indicador de progreso.
- Si ha habido fallo en la conexión, se muestra un mensaje de error.

```
child: FutureBuilder(
  future: _getImage(context, imageUrl),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.done) {
      return CircleAvatar(
        backgroundColor: Colors.black,
        radius: radius + 2.0,
        child: CircleAvatar(
          radius: radius,
          backgroundImage: snapshot.data == null
              ? AssetImage('images/avatar_boy.png')
              : snapshot.data.image,
          backgroundColor: Colors.white,
        ),  // CircleAvatar
      );  // CircleAvatar
    }

    if (snapshot.connectionState == ConnectionState.waiting) {
      return CircularProgressIndicator();
    }

    return Text('Error');
  },
),  // FutureBuilder
```

Figura 4.18: Código - Acceso a imágenes en la base de datos desde la pantalla Acceso de usuario

4.3. Ejemplo de implementación de las pantallas

Una vez implementados todos los componentes y el acceso a la base de datos, ya se pueden pintar con datos reales las pantallas de la aplicación, obteniendo así el resultado final tan esperado.

En esta última subsección se pretende explicar el proceso de implementación llevado a cabo para cada una de las pantallas de la aplicación. Para ello, se tomará como ejemplo la pantalla de Mis colecciones. Se ha decidido escoger esta pantalla ya que cuenta con componentes complejos y realiza el acceso a la base de datos para obtener varios tipos distintos de datos.

El primer paso en cada pantalla es declarar las variables que necesitará. En todas aquellas en las que se realiza un acceso a la base de datos, es necesario contar con una variable en la que se guarde una instancia de Firestore, a partir de la cual se realizarán todas las consultas necesarias. Estas consultas pueden requerir información procedente de otras pantallas, es por ello que aquellas que así lo requieran, contarán con un constructor que será llamado por la pantalla desde la cual se navegó a la actual. En la figura 4.19 se pueden observar las variables declaradas para la pantalla de Mis

colecciones. En la variable *userId* se guarda el id del usuario que ha accedido a la aplicación, el cual es necesario incluir en las consultas de esta pantalla. En este caso, la variable *userId* es enviada desde la pantalla Menú principal. En la segunda variable *firestore* se almacena la instancia de Firestore. Una vez declaradas todas las variables necesarias, se emplea el método build para construir la pantalla.

```
class MyCollectionsPage extends StatelessWidget {
  final String userId;
  final _firestore = FirebaseFirestore.instance;

  MyCollectionsPage({@required this.userId});
```

Figura 4.19: Variables de la pantalla Mis colecciones

Todas las pantallas de la aplicación parten del mismo widget, el Scaffold. A través de sus atributos *appbar* y *body*, se puede indicar el encabezado y cuerpo de la misma. Para el encabezado, se utilizará siempre el widget AppBar, al cual se le indica el título y los botones que contendrá. En cuanto al cuerpo, se comienza siempre con el widget SafeArea, que se encarga de insertar a su hijo con suficiente relleno para evitar intrusiones por parte del sistema operativo. Este hijo, será entonces, todo aquello que se quiera mostrar en el cuerpo.

En la pantalla tomada como ejemplo, el hijo de SafeArea es un StreamBuilder. El funcionamiento de este widget para este caso es muy parecido al indicado en la figura 4.14, con la única diferencia de que en este caso se van construyendo componentes CollectionCard para cada una de las colecciones asociadas al usuario.

El resultado final de la pantalla Mis colecciones se muestra en la figura 4.20.

Figura 4.20: Pantalla final de Mis colecciones

5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este TFG se propuso crear una aplicación gamificada para niños con TEA con el objetivo de motivar su camino de vida, fomentar la interacción social con otros compañeros y promover el aprendizaje visual de ciertas materias.

Entre los entornos de desarrollo multiplataforma disponibles, se decidió utilizar Flutter por ventajas que ofrece como su rápido renderizado de vistas y la función de recarga activa, por la facilidad de interpretar su estructuración en widgets, y, sobre todo, porque se buscaba aprender nuevas tecnologías.

El proceso de análisis y diseño comenzó con el estudio del público objetivo de la aplicación, ya que este iba a influir en la toma de decisiones. En la parte de análisis, se completaron el listado de los requisitos funcionales y no funcionales, el listado de los casos de uso y los primeros modelados de datos. En cuanto al diseño, se prepararon bocetos de las pantallas de la aplicación para tener un punto de partida en la implementación del proyecto.

Una vez finalizado este proceso, se pasó al desarrollo de la aplicación. La idea era implementar primero las pantallas con datos hardcodeados, para, a continuación, implementar el acceso a la base de datos y pintarlas ya con datos reales. La estrategia que se siguió en la implementación fue construir primero componentes a partir de otros widgets anidados entre sí, los cuales se reutilizarían en varias pantallas. La razón de ello era ahorrar tiempo y más líneas de código de las necesarias.

La aplicación es funcional estando conectado a Internet. Sin embargo, no se ha llegado a realizar pruebas exhaustivas con un número elevado de usuarios reales, ya que, en casos como este, la carga masiva de imágenes podría llegar a ser un problema.

5.2. Trabajo futuro

Para que la aplicación esté del todo completa y los alumnos y profesores tengan la mejor experiencia posible, sería necesario implementar una interfaz de administrador. De esta forma, los profesores

podrían monitorizar día a día los resultados de sus alumnos, así como asignarles nuevas colecciones o nuevos hitos.

Por último, sería muy interesante y sobre todo muy gratificante probar algún día personalmente la aplicación con los alumnos del Colegio de Educación Especial Alenta. [1]

[1] https://www.alenta.org/colegio

Bibliografía

[1] L. Cadieux and M. Keenan, “Can social communication skills for children diagnosed with autism spectrum disorder rehearsed inside the video game environment of minecraft generalize to the real world?,” *JMIR serious games*, vol. 8, no. 2, p. e14369, 2020.

[2] A. A. Navan and A. Khaleghi, “Using gamification to improve the education quality of children with autism,” *Revista científica*, no. 37, pp. 90–106, 2020.

[3] “React native - a framework for building native apps using react..” `https://reactnative.dev/`. Último acceso: junio 2021.

[4] “Ionic framework - one codebase. any platform..” `https://ionicframework.com/`. Último acceso: junio 2021.

[5] “Flutter - beautiful native apps in record time.” `https://flutter.dev/`. Último acceso: junio 2021.

[6] “Dart programming language.” `https://dart.dev/`. Último acceso: junio 2021.

[7] “Cloud firestore | firebase - google.” `https://firebase.google.com/docs/firestore?hl=es`. Último acceso: junio 2021.

[8] “Google cloud storage.” `https://cloud.google.com/storage?hl=es-419`. Último acceso: junio 2021.

Apéndices

A

Tablas de casos de uso

Tabla A.1: Caso de uso 01 - Acceso estudiante

CU-01	Acceso esudiante
Versión	1.2 (25/03/2021)
Actor	Estudiante
Precondición	-
Descripción	El estudiante accederá al sistema como se describe en el flujo básico.
Flujo básico	
1	Tras abrir la aplicación, el sistema solicita al estudiante pulsar en su tarjeta de usuario.
2	El estudiante pulsa en su foto.
Flujo alterno	
-	-
Postcondición	El usuario ingresa al sistema con su cuenta de usuario.
Comentarios	Las tarjetas de usuario tienen una posición fija para cada alumno. Caben 3 alumnos por pantalla, si se supera el número, hay scroll horizontal para pasar al siguiente grupo de tarjetas.

Tabla A.2: Caso de uso 02 - Ver colecciones

CU-02	Ver listado de colecciones
Versión	1.1 (31/01/2021)
Actor	Estudiante
Precondición	El estudiante ha accedido a la aplicación con su cuenta de usuario.
Descripción	El estudiante puede ver todas sus colecciones listadas. La información que se muestra para cada colección es la siguiente: nombre, imagen y número de cromos obtenidos por el usuario y número de cromos total de la colección.

Flujo básico	
1	El estudiante pulsa en la sección de "Mis colecciones".
2	El sistema muestra el listado de las colecciones del sistema.
Flujo alterno	
2	Si el usuario no tiene colecciones asociadas, el sistema muestra por pantalla el mensaje "No tienes colecciones"
Postcondición	El estudiante observa por pantalla un listado de sus colecciones.
Comentarios	Texto o dibujo de "No tienes colecciones". Si el estudiante tiene todos los cromos de una colección, el texto que muestra el número de cromos obtenidos y total de la colección, cambiará a color verde.

Tabla A.3: Caso de uso 03 - Ver colección

CU-03	Ver colección
Versión	1.0 (20/01/2021)
Actor	Estudiante
Precondición	El estudiante ha pulsado en "Mis coleccionesz puede ver el listado de sus colecciones.
Descripción	El estudiante podrá más en detalle cada una de sus colecciones.
Flujo básico	
1	El estudiante selecciona la colección que desea ver.
2	El sistema muestra por pantalla los detalles de la colección seleccionada, incluyendo los cromos que la componen.
Flujo alterno	
2	Si el usuario no tiene colecciones asociadas, el sistema muestra por pantalla el mensaje "No tienes colecciones"
Postcondición	El estudiante observa por pantalla los detalles de la colección que ha seleccionado.
Comentarios	Scroll horizontal para la parte de los cromos que componen la colección. Si el estudiante tiene desactivados los textos en la aplicación, la descripición de la aplicación no aparecerá.

Tabla A.4: Caso de uso 04 - Ver cromo obtenido

CU-04	Ver cromo obtenido de una colección
Versión	1.0 (20/01/2021)
Actor	Estudiante

Precondición	El estudiante ha seleccionado una colección del listado de colecciones.
Descripción	Dentro de una colección, el estudiante podrá ver los detalles de los cromos que ya ha obtenido. Esto incluye, su foto y su descripción.
Flujo básico	
1	El estudiante pulsa sobre el cromo ya obtenido que desea revisar.
2	El sistema muestra los detalles del cromo seleccionado.
Flujo alterno	
-	-
Postcondición	El estudiante observa por pantalla los detalles de un cromo que ya tiene.
Comentarios	Si el estudiante tiene desactivados los textos en la aplicación, la descripición del cromo no aparecerá.

Tabla A.5: Caso de uso 05 - Ver cromo no obtenido

CU-05	Ver cromo no obtenido de una colección
Versión	1.1 (31/01/2021)
Actor	Estudiante
Precondición	El estudiante ha seleccionado una colección del listado de colecciones.
Descripción	Dentro de una colección, el estudiante podrá ver las posibilidades que tiene para adquirir cromo que no posee.
Flujo básico	
1	El estudiante pulsa sobre el cromo no obtenido que desea adquirir.
2	El sistema muestra las dos opciones que hay para adquirir dicho cromo: mediante intercambio por puntos o mediante intercambio con otro estudiante.
Flujo alterno	
-	-
Postcondición	El estudiante observa por pantalla las opciones que tiene para adquirir el cromo.
Comentarios	Ambas opciones vendrán representadas por el pictograma correspondiente. En el caso de intercambio por puntos se indica también el número de puntos necesarios para adquirirlo.

Tabla A.6: Caso de uso 06 - Obtener nuevo cromo mediante intercambio por puntos

CU-06	Obtener nuevo cromo mediante intercambio por puntos
Versión	1.0 (20/01/2021)
Actor	Estudiante
Precondición	El estudiante ha seleccionado un cromo que no posee.
Descripción	El estudiante puede obtener un nuevo cromo comprándolo con sus puntos.
Flujo básico	
1	El estudiante pulsa sobre la opción de adquirir nuevo cromo mediante intercambio por puntos.
2	El sistema muestra un mensaje de confirmación si el estudiante tiene suficientes puntos.
3	El estudiante confirma la compra del cromo.
Flujo alterno	
2	Si el estudiante no tiene suficientes puntos, se muestra un mensaje informándole de ello al estudiante.
3	Si el estudiante rechaza la compra, el mensaje de confirmación se cierra sin navegar a otra página.
Postcondición	El estudiante adquiere el nuevo cromo.
Comentarios	Si el estudiante pulsa fuera de cualquier mensaje de confirmación o informativo, el mensaje también se cierra.

Tabla A.7: Caso de uso 07 - Ver hitos

CU-07	Ver hitos
Versión	1.2 (25/03/2021)
Actor	Estudiante
Precondición	El estudiante ha accedido a la aplicación con su cuenta de usuario.
Descripción	El estudiante podrá ver los detalles de cada uno de sus hitos. Hay dos pestañas para los dos tipos de hitos: pendientes por completar y completados. En los pendientes por completar, se muestra el pictograma correspondiente, la descripción y la recompensa en puntos por completarlo. En aquellos ya completados, en lugar de la recompensa se muestra la fecha en la que se completó.
Flujo básico	
1	El estudiante pulsa en la sección de "Mis objetivos".

2	El sistema muestra el listado de los hitos del usuario que están por completar.
Flujo alterno	
-	-
Postcondición	El estudiante observa por pantalla un listado de sus hitos.
Comentarios	El estudiante puede cambiar entre el listado de hitos por completar y completados pulsando en la pestaña correspondiente. Dentro de un mismo listado, scroll horizontal para ir de un hito a otro.

Tabla A.8: Caso de uso 08 - Obtener puntos al completar un hito

CU-08	Obtener puntos al completar un hito
Versión	2.0 (10/05/2021)
Actor	Estudiante
Precondición	El estudiante ha completado un hito que tenía por completar.
Descripción	Cuando un estudiante complete un hito, el profesor o profesora lo marcará como completado desde la propia cuenta del estudiante a través de un código.
Flujo básico	
1	El profesor pulsa sobre el hito que ha completado el alumno.
2	El sistema pregunta al profesor por un código.
3	El profesor introduce el código.
4	El hito se marca como completado y el estudiante obtiene la recompensa en puntos correspondiente.
Flujo alterno	
3	Si el profesor introduce un código erróneo, se le permite volver a intentarlo.
Postcondición	El hito recién completado pasa al listado de hitos completados (añadiendo en este momento la fecha en la que se completó) y el estudiante obtiene la recompensa.
Comentarios	El profesor puede cancelar el proceso de introducir el código.

Tabla A.9: Caso de uso 09 - Proponer intercambio de cromos

CU-09	Proponer intercambio de cromos
Versión	1.1 (31/01/2021)
Actor	Estudiante

Precondición	El estudiante ha seleccionado un cromo que no posee.
Descripción	El estudiante puede proponer un intercambio a otro estudiante para obtener un cromo que no posee.
Flujo básico	
1	El estudiante pulsa sobre la opción de adquirir nuevo cromo mediante intercambio con estudiante.
2	El sistema muestra un listado de los estudiantes a los que proponer un intercambio.
3	El estudiante selecciona el compañero con el que quiere intecambiar cromos.
4	El sistema muestra un mensaje de confirmación.
5	El estudiante confirma la propuesta de intercambio con el compañero seleccionado.
Flujo alterno	
5	Si el estudiante rechaza la propuesta de intercambio, el mensaje de confirmación se cierra sin navegar a otra página.
Postcondición	Se crea un nuevo intercambio entre ambos estudiantes.
Comentarios	El estudiante puede cancelar el proceso de proponer intercambio en cualquier momento. Si el estudiante pulsa fuera de cualquier mensaje de confirmación, el mensaje también se cierra.

Tabla A.10: Caso de uso 10 - Ver peticiones de intercambio recibidas

CU-10	Ver propuestas de intercambio de cromos
Versión	1.0 (20/01/2021)
Actor	Estudiante
Precondición	El estudiante ha accedido a la aplicación con su cuenta de usuario.
Descripción	El estudiante podrá ver las propuestas de intercambio que ha recibido de otros compañeros. En cada una de ellas se muestra el nombre del compañero que ha propuesto el intercambio y su foto.
Flujo básico	
1	El estudiante pulsa en la sección de "Mis intercambios".
2	El sistema muestra el listado de las propuestas de intercambio recibidas.
Flujo alterno	
2	Si el usuario no tiene propuestas de intercambio, el sistema muestra por pantalla el mensaje "No tienes propuestas de intercambio"

Postcondición	El estudiante observa por pantalla un listado de las propuestas de intercambio que ha recibido.
Comentarios	Texto o dibujo de "No tienes propuestas de intercambio".

Tabla A.11: Caso de uso 11 - Aceptar intercambio

CU-10	Aceptar propuesta de intercambio
Versión	2.0 (25/03/2021)
Actor	Estudiante
Precondición	El estudiante ha pulsado en Mis intercambios y puede ver el listado de las propuestas de intercambio recibidas.
Descripción	Para completar el intercambio, el estudiante selecciona el cromo que quiere obtener a cambio.
Flujo básico	
1	El estudiante acepta la propuesta de intercambio pulsando en el botón correspondiente.
2	El sistema muestra la información del intercambio. Esto incluye, los nombres y fotos de ambos estudiantes y el cromo que seleccionó el primero de ellos.
3	El estudiante comienza el proceso para seleccionar el nuevo cromo que obtendrá al finalizar el intercambio.
4	El sistema muestra al estudiante todos los posibles cromos que puede obtener a cambio.
5	El estudiante selecciona el cromo que desea obtener a cambio.
6	La pantalla de información del intercambio se actualiza con el cromo seleccionado.
7	El estudiante pulsa en el botón de completar intercambio.
8	El sistema muestra un mensaje de confirmación.
9	El estudiante confirma el intercambio.
Flujo alterno	
7	Si el usuario cambia de idea, puede pulsar en su cromo para seleccionar una distinto, repitiéndose así el flujo desde el punto 4.
9	Si el estudiante rechaza el intercambio, el mensaje de confirmación se cierra sin navegar a otra página.
Postcondición	Ambos estudiantes involucrados en el intercambio, reciben el cromo que han escogido.

Comentarios	En la pantalla de información del intercambio, si el estudiante no ha seleccionado un cromo todavía, aparece en su lugar un icono de añadir cromo. Además, mientras no haya un cromo seleccionado, no se podrá pulsar en el botón de completar intercambio. Este botón será de color gris mientras esté deshabilitado y de color verde mientras si lo esté.

Tabla A.12: Caso de uso 12 - Cambiar ajustes

CU-11	Cambiar ajustes de usuario
Versión	1.0 (20/01/2021)
Actor	Estudiante
Precondición	El estudiante ha accedido a la aplicación con su cuenta de usuario.
Descripción	El estudiante puede cambiar los ajustes de usuario en el menú de ajustes. Con los ajustes de usuario se puede cambiar el color de la aplicación y habilitar o deshabilitar algunos textos de la aplicación.
Flujo básico	
1	El estudiante pulsa en el botón de "Mis ajustes".
2	El sistema muestra los ajustes de usuario.
3	El estudiante cambia sus ajustes de usuario.
Flujo alterno	
-	-
Postcondición	Se aplican los cambios realizados en los ajustes de usuario.
Comentarios	-


UAM
UNIVERSIDAD AUTONOMA
DE MADRID